NTU Vote : Attack and Defense

A system-wide overview

Chia-You Chen
Department of Computer Science
NTU
Taiwan
b04611015@ntu.edu.tw

Wei-Hsuan Chiang
Department of Computer Science
NTU
Taiwan
b04902077@ntu.edu.tw

Liang-Kuan Chou
Department of Computer Science
NTU
Taiwan
b04902060@ntu.edu.tw

Yen-Min Hsu
Department of Electrical Engineering
NTU
Taiwan
b03901027@ntu.edu.tw

ABSTRACT

In this report, we investigate the e-voting system called "NTU Vote Ver2.0". Our goal is to figure out potential weaknesses. We first introduced how this system works. Then, we researched for CVE online based on the developed framework. We also discovered a random generator used by the system is predictable. At last, we discussed the anonymity, which is quite vital to an e-voting system.

KEYWORDS

E-Voting, Anonymity, MIFARE, CSPRNG

1 INTRODUCTION

Since long time ago, voting has always been a common, important, and irreplaceable way to realize the spirit of democracy. With the improvement of technology, electronic voting system is developed to reduce paper usage, save counting time, prevent human-made mistakes, etc. Countries like Switzerland and Estonia have already developed their own e-voting system. However, there might be risks while using e-voting system, such as identity-fraud and cyber-attacks, depending on the implementation of systems.

In this report, we're investigating an e-voting system called "NTU Vote Ver2.0", which was intended for the annual election of the president and student representatives of NTU Student Council. This system is a pioneer in Taiwan since there wasn't any e-voting system with such large eligible voters (over 30,000 students) in Taiwan before. Our goal was to verify the robustness of this system in order to improve it and utilize it in further usages.

First, we figured out how voters vote and how the system configured. Then, we reviewed and tested its open-source code, trying to find out any potential safety issues. We also try to find loopholes in hardware since the system used Mifare to verify identification, which was pretty weak against attacks. In addition, anonymity is a critical issue for voting process, so we would discuss as an independent part at last.

2 SYSTEM OVERVIEW

Consider a voter who wants vote, what he should do is to go to one of stations. There will be staffs to check identity by his student ID card. If there's no problem, system will assign a booth to him, and he will vote at that booth.

As to the aspect of the system, it is more complicated.. For each stations, there is an independent computer running NTU Vote Ver2.0. These computers are connected with the same central server, which is called "auth server". They are also connected with their own tablets as booths.

If a voter, Bob, went to station A to vote, as mentioned above, staffs would check his identity by student ID card. NTUvote would send a packet to auth server and it would pass the packet to school server to get Bob's information. Then auth server send the information back to station A to let staffs verify the identity. If the identity were correct, auth server would generate an "authcode" corresponded to Bob in its database. This authcode was critical to this system since all the information stored in database could only be accessed by it.

On the other hand, when Bob's authcode was generated, station A would assign an idle booth to him. After Bob finished voting, the booth would send the ballot to the station. The station would collect ballots and send them to auth server. The ballots would then store in the database. Fig(1) shows the relationship between different machines.

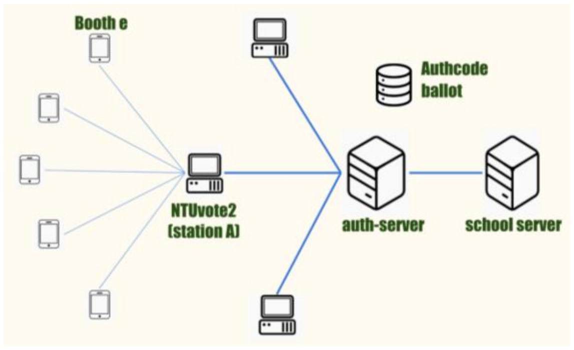

Fig. 1 System Configuration

Two important factors establish the security of this system. One is authcode and the other is HTTPS. Authcode is the key to the database. Therefore, the difficulty of guessing an authcode decides the security of the database. Another potential weakness is the packet sending through the Internet. HTTPS protects packets from attacks such as eavesdropping and man-in-the-middle. For fundamental safety, NTUvote seems to be well enough. However, we will discuss some potential loophole of the system.

3 PROBLEM STATEMENT

While holding an election, there are lots of things we need to take care of. To begin with, we need to make sure that the system is robust. Any little flaw in the system would lead to tremendous damage. For example, if the system shutdown during the election, there would need to have another election and cost lots of money. On top of that, the generation of the Authcode need to be as random as possible. If the generator isn't random enough, someone could fake the Authcode and make unauthorized ballot. Last but not least, the importance of "anonymity" cannot be overemphasized. Once the anonymity is compromised, lots of malicious activities may occur and the fairness of the election can be severely threatened. For instance, bribe can take advantage of the leak of voters' identity to prove if their bribe successfully affected the victim's vote decision, so that the bribers will have more incentive to commit such crime than the anonymity is well-accomplished. In addition, once the decision of a voter can be known by some eavesdroppers or people with bad intention, voters may face some malicious intimidations. To check if such property is compromised, we researched on the procedures and implementations of student elections in NTU.

4 RESULTS

4.1 NTU VoteV2

There are six CVEs about Django 1.9.5 on the database.

CVE-2017-7234

A maliciously crafted URL to a Django (1.10 before 1.10.7, 1.9 before 1.9.13, and 1.8 before 1.8.18) site using the django.views.static.serve() view could redirect to any other.

Essential : Google analysis

CVE-2017-7233

Django 1.10 before 1.10.7, 1.9 before 1.9.13, and 1.8 before 1.8.18 relies on user input in some cases to redirect the user to an "on success" URL. The security check for these redirects (namely ``django.utils.http.is_safe_url()``) considered some numeric URLs "safe" when they shouldn't be, aka an open redirect vulnerability. Also, if a developer relies on ``is_safe_url()`` to provide safe redirect targets and puts such a URL into a link, they could suffer from an XSS attack.

Essential : is_safe_url()

CVE-2016-9014

Django before 1.8.x before 1.8.16, 1.9.x before 1.9.11, and 1.10.x before 1.10.3, when settings.DEBUG is True, allow remote attackers to conduct DNS rebinding attacks by leveraging failure to validate the HTTP Host header against settings.ALLOWED_HOSTS.

Essential: settings.DEBUG is True

CVE-2016-9013

Django 1.8.x before 1.8.16, 1.9.x before 1.9.11, and 1.10.x before 1.10.3 use a hardcoded password for a temporary database user created when running tests with an Oracle database, which makes it easier for remote attackers to obtain access to the database server by leveraging failure to manually specify a password in the database settings TEST dictionary.

Essential: test mode with Oracle Database

CVE-2016-7401

The cookie parsing code in Django before 1.8.15 and 1.9.x before 1.9.10, when used on a site with Google Analytics, allows remote attackers to bypass an intended CSRF protection mechanism by setting arbitrary cookies.

Essential: Google Analysis

CVE-2016-6186

Cross-site scripting (XSS) vulnerability in the dismissChangeRelatedObjectPopup function in contrib/admin/static/admin/js/admin/RelatedObjectLookups.js in Django before 1.8.14, 1.9.x before 1.9.8, and 1.10.x before 1.10rc1 allows remote attackers to inject arbitrary web script or HTML via vectors involving unsafe usage of Element.innerHTML.

Essential: Django CMS

To conclude, only CVE-2016-9014 may be a possible vulnerability on the system. We could utilize it to launch a DNS rebinding attack to avoid the check of Same-origin policy. Then, lots of other attack could be implemented.

4.2 Mercurius System

The verification flow of the election system:

1. Put the Student ID card on the Reader
2. Reader Send Request to Academic Affairs' API
3. Reader Get Student Information Response from the API
4. Scrutineer checks the student identity

In the verification flow, we could see that there are several trivial attacks. To begin with, if you are so fortunate that you find someone's ID card, then you could disguise one's looks and vote for the person. Moreover, if the man who authorized your student ID is compromised by you, then the mission will be much easier. By the way, the two factors aren't impossible, since we could see lots of posts with student ID cards loss at the FB site of NTU or the NTU board on PTT. For the latter factor, since the recruit of electoral workers aren't rigorous enough; moreover, it has almost no requirements and judgements.

However, there is a more serious and fundamental problem in the verification flow. The verification media, NTU student ID card, which is based on MIFARE CLASSIC 1000 were totally compromised in 2007. First of all, the card is only a memory card. Namely, it's protection is only depends on encryption of message on it. However, the encrypted key has only 48 bits long. It's quite huge for scenario that each card uses different key. Unfortunately, the card uses the same key with all NTU student Cards. And the information in the encrypted sector is the student ID. As a result, if we get the key, we could fake anyone's student ID card with only the information of his student ID.

The primary attacks for MIFARE CLASSIC 1000 are PCD-based attack and Sniff-based attack. The PCD-based attack is an offline method to crack. Briefly speaking, the PCD-based attack is just a brute-force guessing. Because the key is 48 bits long, we could crack a single key within three days with the help of GPU. If the key isn't shared, this might be long enough to avoid disastrous fake. The other method is sniff-based, which is an online attack. It sniffed the message and do some replay attack to the reader. However, it's not suitable to this case. Since it's an online method, we need to get the reader. Generally, the reader isn't easy to get.

4.3 NTU-auth Server

Auth-server uses codeigniter as its php framework, and its php version is 7.0, running on Apache2.

4.3.1 Analysis on the Authcode

An authcode is equivalent to a unique ballot which can ideally only be used and accessed by the corresponding voter. Nevertheless, after the examination, we found some weakness in the Authcode caused by the generating process.

4.3.1.1 The generation of Authcode

Authcode is composed of three parts. The first part of it is the state and the type of a ballot, and the second part is made up of the first part, two "randomly generated" strings, and the result of some hash procedure of these two strings. Finally, the third part is the hash of the second part. What the server will put in database for following verification is the third part and the first part.

4.3.1.2 Predicting the Authcode

The program uses the self-defined function random_string() [1] containing the php built-in function mt_rand(), [2] which was wrongly implemented until php 7.1.0, to generate the two random strings. [3] Before php 7.1.0, one can know all the sequence produced by the function once he/she knows the seed used in mt_rand(). After php 7.1.0, the function is implemented with the standard implementation of Mersenne Twister, or "mt19937". However, the story is not over yet. Observing a sufficient number of iterations (624 in the case of MT19937, since this is the size of the state vector from which future iterations are produced) allows one to predict all future iterations.

In conclusion, to predict part of the Authcode, the attacker should know the seed of mt_rand(), (for php version less than 7.1.0) or he/she can observe an enough number of used Authcodes. (for php version greater equal than 7.1.0) Moreover, if the attacker also knows the order and the identities of a specific voter, then a valid Authcode made by the attacker is possible.

4.3.2 Advice for Improvement

The main problem is that neither the wrongly implemented mt_rand() or the right one is a cryptographically secure pseudo-random number generator (CSPRNG). A CSPRNG should satisfy the next-bit test and withstand "state compromise extensions". The former is that if any attacker who knows the first K bits (but not the seed) cannot predict the $(k+1)^{st}$ bit with reasonable computational power, and the latter is that if part or all of the generator's state has been revealed (or guessed correctly), it should be impossible to reconstruct the stream of random numbers prior to the revelation. [4]

Given these facts, when it comes to generating an unpredictable random string such as the Authcode, the designer should use a CSPRNG instead of a PRNG to improve the unpredictability.

4.4 Anonymity Analysis

As for the anonymity part, we discovered two major potential vulnerabilities: "small anonymity set," and "git commit log."

4.4.1 Small Anonymity Set

The vote rate in NTU is often small (on average 4-8%, as shown in Fig(2)). As a result, the anonymity set could be quite small, especially for those voters of Student Council representatives of small colleges. According to the election report[4], the minimum anonymity set can be as few as only tens of people. As a result, the attackers can spend less time to search for their true identity and find out their decision. The reduction of efforts needed to brute force search shows threats to the anonymity property. With such characteristic of NTU election and the release of Git commits of ballot pools, the anonymity corner of the CIA triangle is not highly protected. However, this is not a technical, but an essential problem. The reason lies in the low vote rate and the existence of small college. Take real world as example, there are maybe one or two aboriginal representative voter in a single polling station, which will definitely leak their vote decision. The solution is to gather all the voters to only a few polling stations so that the anonymity set can be larger than original distribution. As for NTU, there are also some possible solutions. First is to promote the vote rate. Once the vote rate is high enough, the anonymity set can be large enough to secure such property. Another is to reduce the number of polling stations. In the most recent election, there are 11 polling stations and voters can freely choose the place to do their votes [6,16,17,18,19]. This once again reduce the size of the anonymity set since those remote stations will have less voters. If those remote stations are eliminated, it will be a great assistance to improve the anonymity property and prevent it from being compromised.

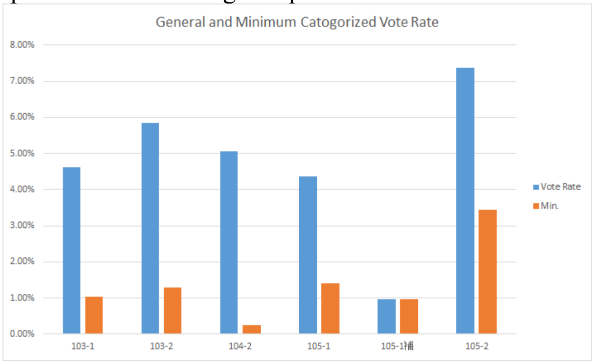

Fig. 2 General and Minimum Categorized Vote Rate

4.4.2 Git Commit Log

This system uses Git commits to collect and synchronize all ballot pools of different polling places. The commit logs will be released with the final election results. Each commit entry represents one vote. It includes a timestamp and the decision. Also, git log will show where the vote is done. These records need to be released to gain trust from the voters. The voters can check whether their vote is really counted into the pool by their voting places and timestamp. However, the timestamp and location information may leak the identity of a voter. The anonymity feature can be easily broken through observing and recording all 11 ballot places' status. For example, if a voter finished his vote at 2015.05.28 09:07:39 with voting for candidate 5, the entry will be "2015.05.28 09:07:39 5." As a result, the bad guys can check their own records to find who did a vote at exactly 2015.05.28 09:07:39 among all the ballot places, and then the voter's decision is leaked. Such shortage is much more threatening than the small anonymity set. The identity-leaking information offers the guys with bad intention a great approach to easily trace every voter's decision through the commit logs. The anonymity is accordingly fully compromised as a cost of the voters' confirmation. One solving approach is applying The ThreeBallot Voting System proposed by Ronald L. Rivest[14] to make vote records with other safer identifier instead of timestamp. However, if there are more than one decision is done on a single voting slip (e.g. for each candidate, make a decision on agree or disagree), the ThreeBallot System is proved to be not secure[15]. As a result, how to strike a balance between anonymity and trust of voters with electronic vote still leave as a significant issue and problem to us and the Election Committee.

5 RELATED WORK

Last year a team in this course studied four e-voting systems in terms of trustiness, verification, concealment, and vulnerability of DoS.[10] The four system they chose are Estonia's system, Helios, a system proposed in 2015 in this course, and their own system, in which they didn't take unstable internet connection into consideration. Compared with their work, we aim to focus on a practical system and take code review level analysis while the last year team restricted themselves to protocol level examination.

[11] provides a paradigm for e-voting system examination. The researchers conducted in-person observation, developer and officials interview, source code evaluation, and attack experiments from client side and server side. (attack against the reproduction of e-voting system in their lab) However, due to the prime minister's and President's mistrust of the researchers, the advice to stop using the system was not accepted. When it comes to our case, our team members are familiar with the developers, and one of our team member is a representative in the student council of NTU, thus it would be much easier to call for improvement if we get specific conclusions from our research.

[12] and [13] focus on the verification mechanism which has not yet been implemented in NTU's e-voting system, but they can be viewed as precedents for the possible voting procedure in the future.

[14] proposed a new ballot protocol to offer the e-system a more secure and anonymous way to implement the voting procedure, while [15] proved that under some circumstances, [14] also had some vulnerabilities against

anonymity and integrity. Their theory and discussion can serve as an alternative way to solve current problems in NTUVote.

6 CONCLUSION AND FUTURE WORK

NTU Vote is a pioneer of e-voting system in Taiwan. Though some of its disadvantages have been raised in this report, it does not mean that this system is a failure. Instead, more efforts should be taken to improve it. Considering realistic condition, building a reliable and usable e-voting system may need not only the ability of writing codes, analyzing the robustness, but also the legality. That is, to push the system into a further implementation, developers should have cross-domain knowledge. This is not an easy task for us. We hope that more other experts engage in development and realize a reliable e-voting system.

REFERENCES

[1] https://github.com/mousems/NTUvoteV2
[2] http://php.net/manual/en/function.mt-rand.php
[3] https://en.wikipedia.org/wiki/Mersenne_Twister#Alternatives
[4] 103-1 學生代表選舉第 3 份選舉公告
[5] 103 學年度第 2 學期臺大學生會暨自治組織聯合選舉
[6] 105 學年度第 2 學期國立臺灣大學學生會暨學生自治組織聯合選舉 第二份選舉公報
[7] Hacking Classic Mifare Card - BLACKHAT 2014
[8] MIFARE Classic *IS* Completely Broken - HITCON 2010 - 鄭振牟
[9] Chen-Lun Huang, Hung-Yu Shen, Hsien-Chun Chiu, Lu-Tzu Li. Secure i-Voting System. 2016.
[10] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, J. Alex Halderman. Security
[11] Analysis of the Estonian Internet Voting System. 2014.
[12] Koksal Mus, Mehmet Sabır Kiraz, Murat Cenk, Isa Sertkaya. Estonian Voting Verification Mechanism Revisited. 2016.
[13] Ivo Kubjas, Tiit Pikma, Jan Willemson. Estonian Voting Verification Mechanism Revisited Again. 2017
[14] Ronald L. Rivest (2006). "The ThreeBallot Voting System"
[15] Charlie E. M. Strauss (2006). "The Trouble with Triples"
[16] 104 學年度第 2 學期國立臺灣大學學生會暨學生自治組織聯合選舉 第二份選舉公報
[17] 104 學年度第 1 學期國立臺灣大學學生會暨學生自治組織聯合選舉 第二份選舉公報
[18] 104 學年度第 2 學期國立臺灣大學學生會暨學生自治組織聯合選舉 第三份選舉公報
[19] 104 學年度第 1 學期國立臺灣大學學生會暨學生自治組織聯合選舉 第三份選舉公報